Introduction à la programmation récursive

Répétition 2

15 octobre 2014

Correction exercices proposés

Exercice 1. Définir un prédicat `insert(+Xs, +E, -Ys)`, vrai si et seulement si la liste `Ys` est la liste `Xs` dans laquelle l'élément `E` a été inséré après chaque élément de `Xs`.

```
?- insert([titanic, a, coule], stop, X).

    X = [titanic, stop, a, stop, coule, stop] ;
    false.
```

Exercice 2. Définir un prédicat `group(+Xs, -Ys)`, vrai si et seulement si la liste `Ys` est la liste des éléments de `Xs`, groupés par listes de deux éléments.

```
?- group([cochon, truie, taureau, vache, cheval, jument], X).

    X = [[cochon, truie], [taureau, vache], [cheval, jument]].
```

Exercices

Exercice 3. Définir un prédicat `sub(+A, +B, +Ls, -Zs)` vrai si la liste `Zs` est la liste `Ls` dans laquelle toutes les occurrences de `A` ont été remplacées par `B`.

Exercice 4. Définir un prédicat `pack(+Ls, -Zs)` vrai si la liste `Zs` est la liste `Ls` dans laquelle les séquences d'éléments identiques ont été remplacés par des sous-listes contenant tous ces éléments.

```
?- pack([a, a, a, a, b, c, c, a, a, d, e, e, e, e], X).

    X = [[a, a, a, a], [b], [c, c], [a, a], [d], [e, e, e, e]] ;
    false.
```

Exercice 5. Définir un prédicat `encode(+Ls, -Zs)` vrai si la liste `Zs` est la liste `Ls` dans laquelle chaque séquence de symboles identiques a été remplacée par une liste de deux éléments.

```
?- encode([a, a, a, a, b, c, c, a, a, d, e, e, e, e], X).

    X = [[4, a], [1, b], [2, c], [2, a], [1, d], [4, e]] ;
    false.
```

Exercice 6. Définir un prédicat `decode(+Ls, -Zs)` vrai si la liste `Ls` est un encodage (comme défini à l'exercice précédent) de la liste `Zs`.

Exercice 7. Définir un prédicat `prime(+X)` vrai si X est un nombre premier.

Exercice 8. Définir un prédicat `plateau(+Ls,-Ps)` vrai si et seulement si `Ps` est la plus longue sous-liste d'éléments identiques de la liste `Ls`.

Exercices proposés

Exercice 9. Définir un prédicat `swap(+A, +B, +Ls, -Zs)`, vrai si la liste `Zs` est la liste `Ls` dans laquelle toutes les occurrences de `A` ont été remplacées par `B` et toutes les occurrences de `B` ont été remplacées par `A`.

Exercice 10. Définir le prédicat `skipN(+N, +M, +Xs, -Ys)` vrai si `Ys` est la liste `Xs` privée de `N` éléments sur `M` (on supprime les N premiers éléments).

Exercice 11. Définir le prédicat `perfect(+N)` vrai si et seulement si `N` est un nombre parfait. Pour rappel, un nombre est parfait si la somme de ses diviseurs (excepté lui-même) rend le nombre.

Exercice 12. Les nombres de Motzkin vérifient notamment les égalités suivantes :

$$M_0 \;=\; M_1 \;=\; 1$$

$$\forall n > 1 : M_n \;=\; \frac{3(n-1)M_{n-2} \;+\; (2n+1)M_{n-1}}{n+2}$$

Écrire une fonction `motz` qui à tout entier naturel n associe M_n.

Exercice 13. Soit $f : \mathbb{N} \to \mathbb{N}$, la fonction définie par :

$$f(n) \;=\; \sum_{i=0}^{n-1} (f(i) + 2 * i) \bmod (n + i + 3)$$

Définir un prédicat `f(+N, -R)` vrai si f(`N`) = `R`.